Ember CLI Addons

Ember CLI?

Ember CLI stands in the very same point that Rails did a decade ago.

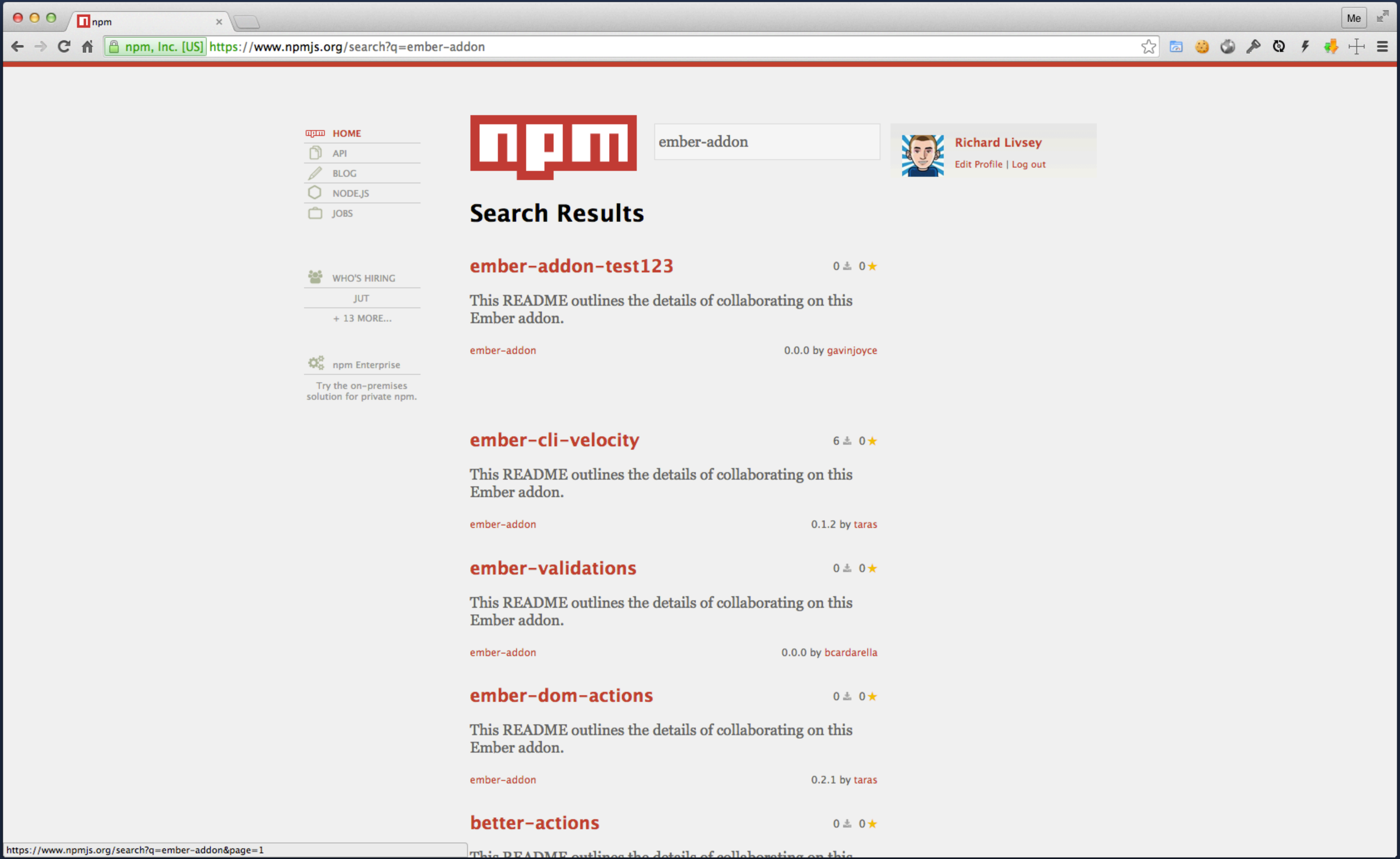
– Miguel Camba - Why I believe that Ember CLI is a game-changer

What can you do with an addon?

- Add routes, controllers, views, components etc…
- Development server middleware
- Blueprints / generators
- Configuration
- Commands
- Modify built assets
- Update build output

Finding Addons

```
// /package.json
{
  "name": "ember-cli-divshot",
  "version": "0.1.4",
  "keywords": [
    "ember-addon"
  ]
}
```



npm

https://www.npmjs.org/search?q=ember-addon

HOME
API
BLOG
NODE.JS
JOBS

WHO'S HIRING
JUT
+ 13 MORE...

npm Enterprise
Try the on-premises solution for private npm.

ember-addon

Richard Livsey
Edit Profile | Log out

Search Results

ember-addon-test123 0 ⤓ 0 ★

This README outlines the details of collaborating on this Ember addon.

ember-addon 0.0.0 by gavinjoyce

ember-cli-velocity 6 ⤓ 0 ★

This README outlines the details of collaborating on this Ember addon.

ember-addon 0.1.2 by taras

ember-validations 0 ⤓ 0 ★

This README outlines the details of collaborating on this Ember addon.

ember-addon 0.0.0 by bcardarella

ember-dom-actions 0 ⤓ 0 ★

This README outlines the details of collaborating on this Ember addon.

ember-addon 0.2.1 by taras

better-actions 0 ⤓ 0 ★

This README outlines the details of collaborating on this

https://www.npmjs.org/search?q=ember-addon&page=1

Installing an Addon

Install it.

```
$ npm install --save-dev ember-sanitize
$ ember generate ember-sanitize
```

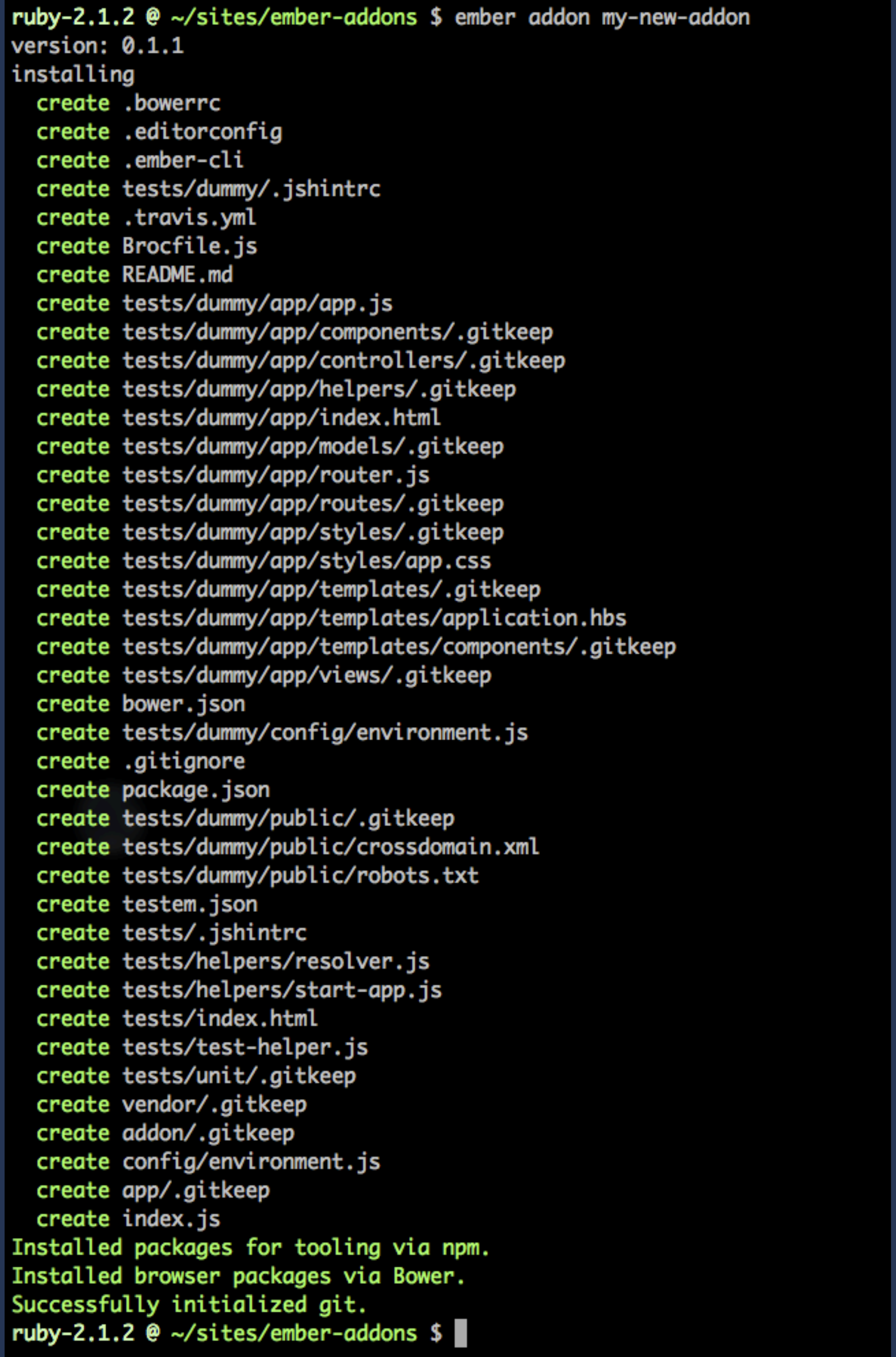
Use it.

```
<h1>
  {{sanitize-html title}}
</h1>
```

Creating an addon

```
$ npm install -g ember-cli

$ ember addon my-new-addon
```



```
ruby-2.1.2 @ ~/sites/ember-addons $ ember addon my-new-addon
version: 0.1.1
installing
  create .bowerrc
  create .editorconfig
  create .ember-cli
  create tests/dummy/.jshintrc
  create .travis.yml
  create Brocfile.js
  create README.md
  create tests/dummy/app/app.js
  create tests/dummy/app/components/.gitkeep
  create tests/dummy/app/controllers/.gitkeep
  create tests/dummy/app/helpers/.gitkeep
  create tests/dummy/app/index.html
  create tests/dummy/app/models/.gitkeep
  create tests/dummy/app/router.js
  create tests/dummy/app/routes/.gitkeep
  create tests/dummy/app/styles/.gitkeep
  create tests/dummy/app/styles/app.css
  create tests/dummy/app/templates/.gitkeep
  create tests/dummy/app/templates/application.hbs
  create tests/dummy/app/templates/components/.gitkeep
  create tests/dummy/app/views/.gitkeep
  create bower.json
  create tests/dummy/config/environment.js
  create .gitignore
  create package.json
  create tests/dummy/public/.gitkeep
  create tests/dummy/public/crossdomain.xml
  create tests/dummy/public/robots.txt
  create testem.json
  create tests/.jshintrc
  create tests/helpers/resolver.js
  create tests/helpers/start-app.js
  create tests/index.html
  create tests/test-helper.js
  create tests/unit/.gitkeep
  create vendor/.gitkeep
  create addon/.gitkeep
  create config/environment.js
  create app/.gitkeep
  create index.js
Installed packages for tooling via npm.
Installed browser packages via Bower.
Successfully initialized git.
ruby-2.1.2 @ ~/sites/ember-addons $
```

What goes where?

- **addon** - */addon*
- **app** - */app*
- **styles** - */app/styles*
- **templates** - */app/templates*
- **vendor** - */vendor*
- **test-support** - */test-support -> /test*
- **public** - */public -> /dist/{addon-name}*

```
// /addon/utils/sanitize.js

function sanitizeElement(element, config) {
  var sanitizer = new Sanitize(config);
  var cleaned   = document.createElement('div');

  cleaned.appendChild(sanitizer.clean_node(element));
  return cleaned.innerHTML;
}

function sanitize(html, config) {
  var container = document.createElement('div');
  container.innerHTML = html;
  return sanitizeElement(container, config);
}

export {
  sanitize
};
```

```
// /app/helpers/sanitize-html.js

import Ember from 'ember';
import { sanitize } from 'ember-sanitize/utils/sanitize';

export default Ember.Handlebars.
  makeBoundHelper(function(html, configName, options) {

  // ...

  var sanitized = sanitize(html, config);
  return new Ember.Handlebars.SafeString(sanitized);
});
```

Importing Bower Packages

Add a generator which adds the Bower package

```
// blueprints/ember-sanitize/index.js

module.exports = {
  normalizeEntityName: function() {
    // this prevents an error when the entityName is
    // not specified (since that doesn't actually matter
    // to us
  },
  afterInstall: function() {
    return this.addBowerPackageToProject('sanitize.js', '*');
  }
};
```

Importing Bower Packages

Then import the bower JS into the project

```
// index.js

module.exports = {
  name: 'ember-sanitize',
  included: function(app) {
    this._super.included(app);
    this.app.import(app.bowerDirectory + '/sanitize.js/lib/sanitize.js');
  }
};
```

Controlling Addon Namespace

Add `/addon/index.js` to manage the entry module for the addon's namespace.

```
// /addon/index.js
import injectScript from './utils/inject-script';
export default injectScript;
```

allows:

```
import injectScript from 'ember-inject-script';
```

vs

```
import injectScript from 'ember-inject-script/utils/inject-script';
```

Controlling Addon Namespace

```
import Model   from './model/model';
import attr    from './model/attr';
import hasOne  from './relationships/has-one';
import hasMany from './relationships/has-many';
import Store   from './store';
//...

export {
  Model,
  attr,
  hasOne,
  hasMany,
  Store,
  //...
};
```

Controlling Addon Namespace

```
import {
  Model,
  attr,
  hasOne
} from 'fireplace';
```

vs

```
import Model from 'fireplace/model/model';
import attr from 'fireplace/model/attr';
import hasOne from 'fireplace/relationships/has-one';
```

Hooks

```
// /index.js
module.exports = {
  name: 'my-addon',
  blueprintsPath:
  config:
  contentFor:
  included:
  includedCommands:
  postBuild:
  postprocessTree:
  serverMiddleware:
  treeFor:
};
```

Hooks
config

Returns object to merge with app config.

```
module.exports = {
  name: 'ember-cli-content-security-policy',
  config: function(environment /*, appConfig */) {
    var ENV = {
      contentSecurityPolicyHeader: 'Content-Security-Policy-Report-Only',
      contentSecurityPolicy: {
        'default-src': "'none'",
        'script-src': "'self'",
        //...
      }
    }
    if (environment === 'development') {
      ENV.contentSecurityPolicy['script-src'] = ENV.contentSecurityPolicy['script-src'] + " 'unsafe-eval'";
    }
    return ENV;
  }
}
```

Hooks
contentFor

Injects HTML into `index.html`, type can be head or body.

```
module.exports = {
  name: 'live-reload-middleware',
  contentFor: function(type) {
    var liveReloadPort = process.env.EMBER_CLI_INJECT_LIVE_RELOAD_PORT;
    if (liveReloadPort && type === 'head') {
      return '<script src="//localhost:' +
        liveReloadPort +
        '/livereload.js?snipver=1" type="text/javascript"></script>';
    }
  }
}
```

Hooks
include

Receives EmberApp instance.

Generally used to call app.import.

```
module.exports = {
  name: 'ember-sanitize',
  included: function(app) {
    this._super.included(app);
    this.app.import(app.bowerDirectory + '/sanitize.js/lib/sanitize.js');
  }
};
```

Hooks
includedCommands

```
module.exports = {
  name: 'ember-cli-divshot',
  includedCommands: function() {
    return {
      'divshot': require('./lib/commands/divshot')
    }
  }
}
```

```
$ ember divshot push
```

Hooks
postBuild

Receives build results with path to the built directory.

```
module.exports = {
  postBuild: function(results) {
    // do something with results.directory
  }
}
```

Hooks
postProcessTree

Allows manipulating the final generated tree.

```
module.exports = {
  name: 'ember-cli-autoprefixer',
  postprocessTree: function (type, tree) {
    if (type === 'all' || type === 'styles') {
      tree = autoprefixer(tree, this.options);
    }
    return tree;
  }
}
```

Hooks
serverMiddleware

Allows hooking into the development server's Express app.

```
module.exports = {
  name: 'live-reload-middleware',
  serverMiddleware: function(config) {
    var app = config.app;
    var options = config.options;

    app.use(livereloadMiddleware({
      port: options.liveReloadPort
    }));
  }
}
```

Hooks
treeFor

- treeFor *(called for all trees)*
- treeForApp
- treeForStyles
- treeForTemplates
- treeForAddon
- treeForVendor

Testing

Ember CLI generates a dummy app in `/tests/dummy`

Used for integration tests.

Served at http://localhost:4200

Tests at http://localhost:4200/tests

Questions?